subst manual page - Tcl Built-In Commands

tcl.tk/man/tcl/TclCmd/subst.htm

NAME

subst — Perform backslash, command, and variable substitutions

SYNOPSIS

subst ?**-nobackslashes**? ?**-nocommands**? ?**-novariables**? *string*

DESCRIPTION

This command performs variable substitutions, command substitutions, and backslash substitutions on its *string* argument and returns the fully-substituted result. The substitutions are performed in exactly the same way as for Tcl commands. As a result, the *string* argument is actually substituted twice, once by the Tcl parser in the usual fashion for Tcl commands, and again by the *subst* command.
If any of the **-nobackslashes**, **-nocommands**, or **-novariables** are specified, then the corresponding substitutions are not performed. For example, if **-nocommands** is specified, command substitution is not performed: open and close brackets are treated as ordinary characters with no special interpretation.

Note that the substitution of one kind can include substitution of other kinds. For example, even when the **-novariables** option is specified, command substitution is performed without restriction. This means that any variable substitution necessary to complete the command substitution will still take place. Likewise, any command substitution necessary to complete a variable substitution will take place, even when **-nocommands** is specified. See the **EXAMPLES** below.

If an error occurs during substitution, then **subst** will return that error. If a break exception occurs during command or variable substitution, the result of the whole substitution will be the string (as substituted) up to the start of the substitution that raised the exception. If a continue exception occurs during the evaluation of a command or variable substitution, an empty string will be substituted for that entire command or variable substitution (as long as it is well-formed Tcl.) If a return exception occurs, or any other return code is returned during command or variable substitution, then the returned value is substituted for that substitution. See the **EXAMPLES** below. In this way, all exceptional return codes are “caught” by **subst**. The **subst** command itself will either return an error, or will complete successfully.

EXAMPLES

When it performs its substitutions, *subst* does not give any special treatment to double quotes or curly braces (except within command substitutions) so the script

```
set a 44
subst {xyz {$a}}
```

returns “**xyz {44}**”, not “**xyz {$a}**” and the script

```
set a "p\} q \{r"
subst {xyz {$a}}
```

returns “**xyz {p} q {r}**”, not “**xyz {p\} q \{r}**”.

When command substitution is performed, it includes any variable substitution necessary to evaluate the script.

```
set a 44
subst -novariables {$a [format $a]}
```

returns “**$a 44**”, not “**$a $a**”. Similarly, when variable substitution is performed, it includes any command substitution necessary to retrieve the value of the variable.

```
proc b {} {return c}
array set a {c c [b] tricky}
subst -nocommands {[b] $a([b])}
```

returns “**[b] c**”, not “**[b] tricky**”.

The continue and break exceptions allow command substitutions to prevent substitution of the rest of the command substitution and the rest of *string* respectively, giving script authors more options when processing text using *subst*. For example, the script

```
subst {abc,[break],def}
```

returns “**abc,**”, not “**abc,,def**” and the script

```
subst {abc,[continue;expr {1+2}],def}
```

returns “**abc,,def**”, not “**abc,3,def**”.

Other exceptional return codes substitute the returned value

```
subst {abc,[return foo;expr {1+2}],def}
```

returns “**abc,foo,def**”, not “**abc,3,def**” and

```
subst {abc,[return -code 10 foo;expr {1+2}],def}
```

also returns “**abc,foo,def**”, not “**abc,3,def**”.